|  |  |  |  |
| --- | --- | --- | --- |
| A black and white logo  Description automatically generated with low confidence | | INTERNATIONAL TELECOMMUNICATION UNION  **TELECOMMUNICATION** **STANDARDIZATION SECTOR**  STUDY PERIOD 2022-2024 |  |
|  |
| **Team:** | *Hyperion* | | |
| **Title:** | *Hyperion AI - Report on* *ITU WTSA Hackathon 2024* | | |
| **Contact:** | Ndabuye Gideon | | E-mail: [sndabuye@gmail.com](mailto:sndabuye@gmail.com) |

|  |  |
| --- | --- |
| **Abstract:** | This document contains the submission report from team Hyperion towards ITU WTSA Hackathon 2024 for the use case “Real-time Network reliability prediction”. |

# 1. Use case introduction: Real-time Network reliability prediction

Ensuring reliable and efficient communication in diverse network environments, including industrial, vehicular, and general-purpose networks, is essential for a wide range of applications. To address the challenge of maintaining network reliability, a real-time machine learning model can be developed to predict network Quality of Service (QoS) based on factors such as traffic patterns, resource utilization, and environmental conditions. By accurately predicting QoS in real-time, the model can optimize resource allocation, enhance adaptive routing, prevent potential service disruptions, and ultimately improve the overall Quality of Experience (QoE).

This approach aims to optimize network performance across various dynamic and demanding environments, ensuring seamless and reliable communication while preventing failures and improving user satisfaction.

## Key Challenges

* Predicting network QoS in dynamic and complex environments.
* Ensuring the model can process data and make predictions in real-time.
* Integrating the model with existing network management systems.

## Proposed Solution

A real-time ML model can be developed to predict network QoS based on various factors, including traffic patterns, resource utilization, and environmental conditions. The model can then be used to attain the below objectives.

* **Optimize resource allocation**

Allocate resources to critical applications based on predicted QoS.

* **Adaptive routing**

Select optimal routes for data packets based on predicted network conditions.

* **Predictive maintenance**

Identify potential network failures or bottlenecks and take proactive measures to prevent service disruptions.

* **Quality of Experience (QoE) improvement**

Tailor network services to individual users' needs, ensuring that critical applications receive the required QoS.

## Sample scenario

**Phase 1 -** A vehicular network is experiencing increasing congestion due to a large influx of vehicles.

**Phase 2 -** The ML model analyses real-time network data and predicts a significant drop in QoS within the next unit time

**Phase 3 -** The network management system is alerted to the predicted QoS degradation.

**Phase 4 -** The system automatically adjusts resource allocation to prioritize critical vehicular applications, such as emergency vehicle communications.

**Phase 5 -** The QoS degradation is mitigated, and the network continues to operate efficiently.

# 2. Use case requirements

1. It is critical that the ML model should accurately predict network reliability/QoS with minimal error.
2. It is critical that the model should be able to process data and make predictions in real-time, with extremely low latency.
3. It is critical that the model should be able to handle large volumes of data and adapt to changing network conditions.
4. It is critical that the model should be easily integrated with existing network management systems.

# 3. Pipeline design

## Core activities

* **Time series analysis –** analyse the time patterns from the network traffic to get general trends
* **Regression analysis –** making predictions on network reliability using QoS metrics based on input features.
* **Anomaly detection –** detecting unusual network behavior that may indicate potential issues.

## Pipeline

The figure below shows an ideal setup of the pipeline into a dynamic environment. The explanations on the components of the diagram follow.

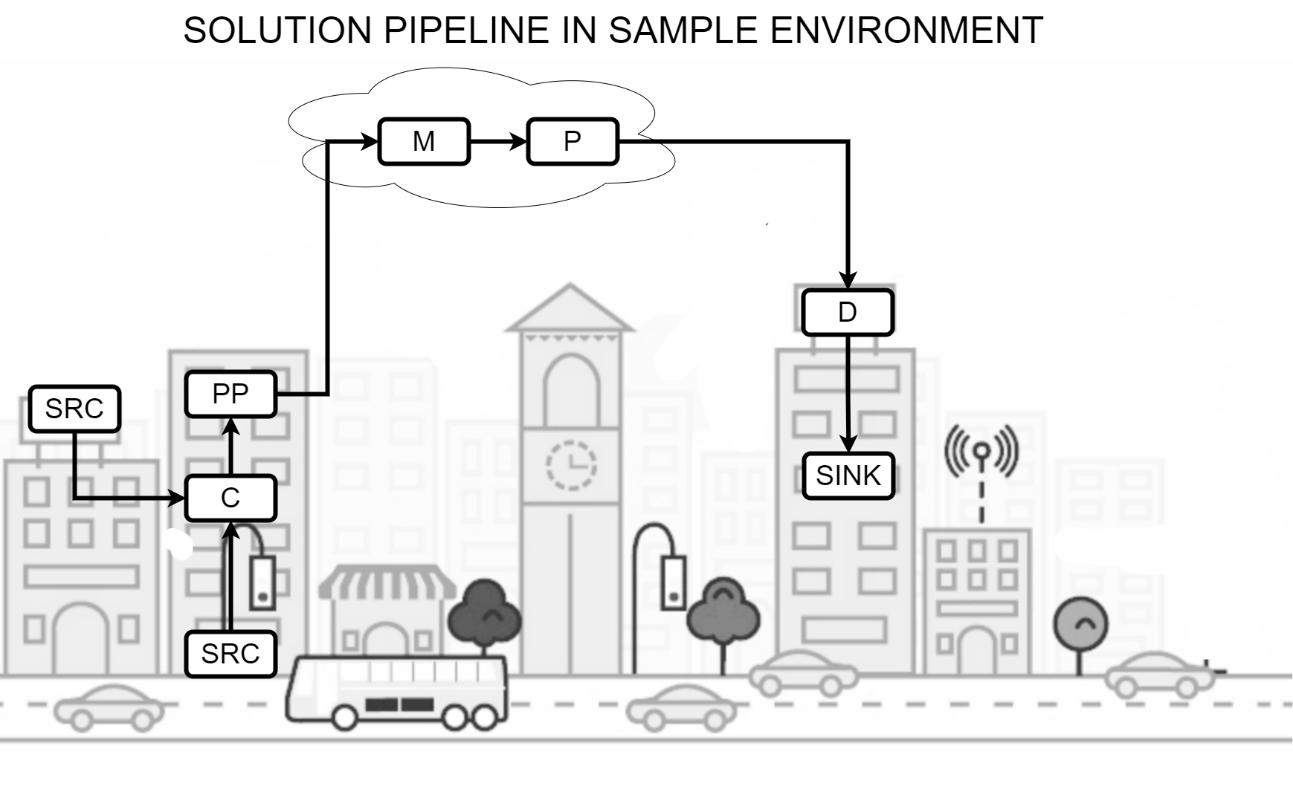
****

Figure 1: Solution pipeline in a sample environment (Dynamic network environment)

1. **SRC (Source)**

Road-side sensors and base stations collect real-time data on traffic conditions and network performance metrics like packet loss and latency.

1. **C (Collector)**

A central server gathers this data from all sources, ensuring it reflects the current state of the network accurately.

1. **PP (Preprocessor)**

The data is cleaned and normalized. Outliers (e.g., faulty sensor readings) are removed, and missing data is imputed. The system then computes rolling averages for features like traffic density and latency over the past unit time.

1. **M (Model)**

A pre-trained time series ML model predicts a QoS drop within the next unit time based on real-time network conditions and historical patterns. For instance, if traffic continues to grow, packet loss will increase, and latency will worsen, leading to degraded network performance. This can be run both in cloud and on-premises.

1. **P (Policy)**

The network applies policies to ensure critical services like emergency vehicle communications are prioritized. Policies dictate that traffic for non-essential services is deprioritized when the network congestion is predicted. This contains predefines set of actions that the system should take based on the results output from the model.

1. **D (Distributor)**

The system distributes the prediction to the network management system and relevant devices in the network (e.g., routers, base stations).

1. **SINK**

Routers and base stations reallocate bandwidth, prioritizing vehicular communications and adjusting routes for packets to mitigate congestion. Emergency communications remain unaffected, while non-critical traffic is delayed or rerouted. This can be a device or software, mostly an xApp configured to take actions within the network.

# 4. Relation to Standards

* **ITU-T Y.317:** Network performance objectives
* **ITU-T Y.321:** Network performance measurement
* **ITU-T Y.350:** Quality of service parameters

# 5. Code submission details

* **Programming language:** Python
* **Libraries:** Scikit-learn, LightGBM
* **Code repository:** [GitHub](https://github.com/nadiez/WTSA-Hyperion)

# 6. Self-Testing results